Regression: want $MSE[f] = E[(Y - f(X))^2]$ to be small

- If optimal regression prediction is $u(x) = E(Y|X=x)$, but $f(y|x)$ is unknown $\left(u(x) = \int_{-\infty}^{\infty} y f(y|x)\,dy\right)$
- Can't just do $E(Y|X=x)$ is $\frac{1}{\#\{i : x_i = x\}} \sum_{i:x_i=x} Y_i$ since that value of x might not appear in the data
- Will use linear smoother so

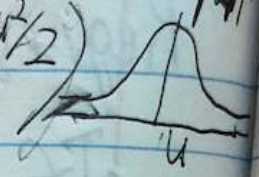
$$\hat{u}(x) = \sum_{i=1}^{n} w(x_i, x)\, Y_i$$

(x with no subscript is the point we are trying to predict)

weight of Y_i depends on what x_i is and on where we're predicting

- common weight functions:
  - global mean $w(x_i, x) = \frac{1}{n}$
  - nearest neighbors $w(x_i, x) = \begin{cases} 1 & \text{if } x_i \text{ is closer to } x \text{ than any other } x_j \\ 0 & \text{otherwise} \end{cases}$
  - k nearest neighbors $w(x_i, x) = \begin{cases} \frac{1}{k} & \text{if } x_i \text{ is one of the } k \text{ closest} \\ 0 & \text{otherwise} \end{cases}$
    - increasing k reduces variance but increases bias
    - for knn: $\hat{u}(x) = \frac{1}{k} \sum_{i:\text{KNN}(x)} u(x_i) + \frac{1}{k} \sum_{i:\text{KNN}(x)} \varepsilon_i$

□ $W(x_i, x) = \frac{k\left(\frac{x_i - x}{h}\right)}{\sum_{j=1}^{n} k\left(\frac{x_j - x}{h}\right)}$ where $k(u)$ is a smooth pdf (say $k(u) = \frac{1}{\sqrt{2\pi}} e^{-u^2/2}$)



h is scale/bandwidth

- data points with $|x_i - x| < h$ matter alot
- data points with $|x_i - x| > h$ hardly matter

The function k is called the kernel and this is kernel regression

- $h \to 0$ looks like nearest neighbors (less bias, more var)
- $h \to \infty$ estimate flattens out to global mean
- $\sum_{i=1}^{n} w(x_i, x, h) = 1$ for kernel regression
- These are nonparametric (global mean, knn, kernel)

Linear regression

A new predicted y is $x_{new}\hat{\beta} = x_{new}(X^T X)^{-1} X^T \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$

$\Rightarrow x_{new}\hat{\beta} = \sum_{i=1}^{n} x_{new} \frac{x_i y_i}{n\hat{\sigma}_x^2}$

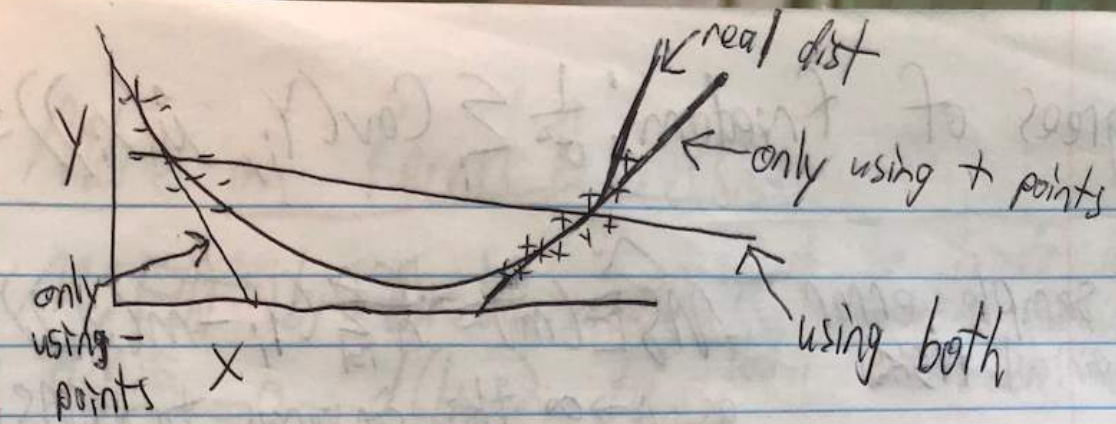
(i) This doesn't depend on x give more weight to points closer to x_{new} and

(ii) $w(x_i, x)$ goes up as x_i moves away from $\bar{x}$

(iii) this only makes sense if u is really linear

- If the true $u(x)$ is not linear the there is always an optimal linear β, but it changes with the distribution of X. Even though best linear model it could still be bad.

Ex:



optimal linear predictor of Y from X: $(\alpha, \beta) = \underset{a,b}{\operatorname{argmin}} E[(Y-(a+bX))^2]$

$\Rightarrow \alpha = E[Y] - \beta E[X]$ and $\beta = \frac{Cov(X,Y)}{Var(X)}$

which says $Cov(X,Y) = \beta Var(X)$

$$\hat{u} = wy = \begin{bmatrix} w(x_1,x_1) & \cdots & w(x_1,x_n) \\ \vdots & \ddots & \vdots \\ w(x_n,x_1) & \cdots & w(x_n,x_n) \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

$\hat{u} = wu + w\epsilon$

$\Rightarrow bias(\hat{u}) = u - wu = (I - w)u$

$\Rightarrow Var(\hat{u}) = Var(w\epsilon) = \sigma^2 w w^T$

assuming that $Var(\epsilon_i) = Var(\epsilon_j) = 0 \ \forall i,j$ and $Cov(\epsilon_i, \epsilon_j) = 0$ (unless $i = j$)

- for linear models $w = X(X^TX)^{-1}X^T = H$ (hat matrix)

degrees of freedom: $\frac{1}{\sigma^2}\sum_i \text{Cov}(Y_i, \hat{\mu}(x_i)) = \text{tr}(W)$

In sample error (empirical ~~MSE~~) $\widehat{MSE}(m) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - m(x_i))^2$

as $n \to \infty$ this converges to $MSE(m)$

- Usually we have $\hat{m}$ estimated from the data (so it's not fixed in advance)
- Minimizing $\widehat{MSE}$ partially reduces MSE but also partially memorizes noise

How to avoid memorizing noise?

a) use theory to estimate/bound $MSE(\hat{m}) - \widehat{MSE}(\hat{m})$ (degrees of freedom adjustments)

b) use new data to evaluate different models (calculate MSE for each candidate model with hold-out/validation data)
- can't use the new data to fit model

c) Fake having new data with cross-validation

k-folds cross validation:

- randomly divide data into k equals parts
- each part gets used once as the testing set and use the other k-1 folds as the training set
- Leave one out cross validation (LOOCV) is using n-folds (only 1 point in testing set)
- After using k-folds to pick the model fit the chosen model to all of the data